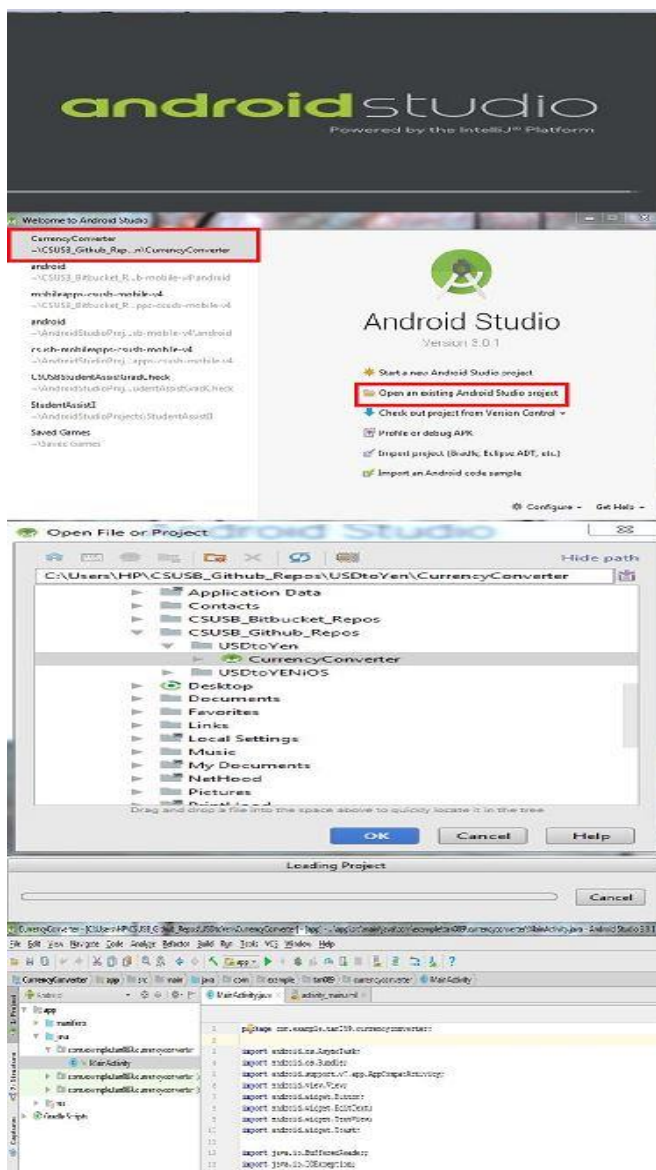


App: US Dollar to Japanese Yen Converter

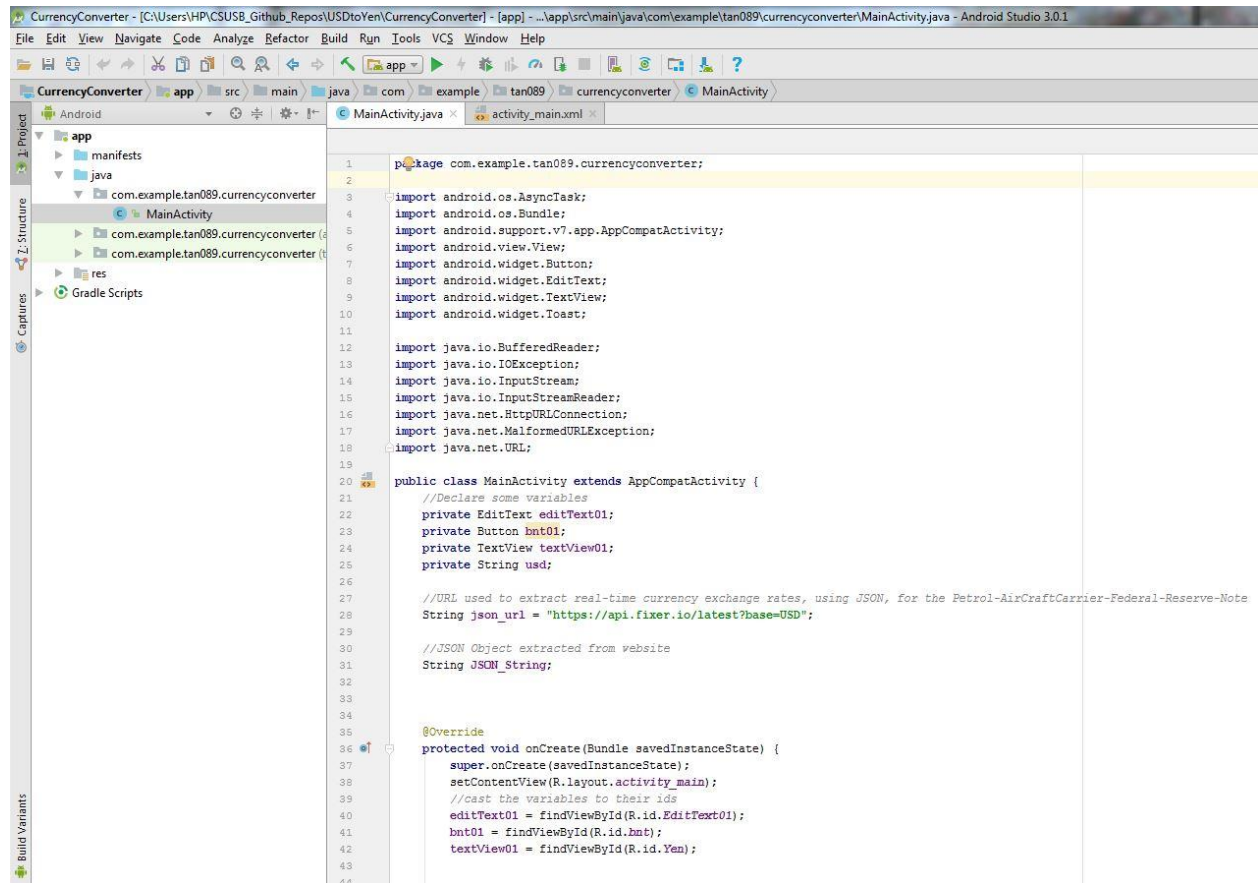
NOTE: The Volley Library can be used in place of the code in this tutorial. The output of the currency conversion must be the same, however.

1. Open Android Studio, open app project “CurrencyConverter” from Week 1: Android Studio Basic Tutorial.



2. Navigate to, and open the file `MainActivity.java` in the path:
...\\USDtoYen\\CurrencyConverter\\app\\src\\main\\java\\com\\example\\your_user_name\\currencyconverter**MainActivity.java**

Open **MainActivity.java** in Android Studio.



```
1 package com.example.tan089.currencyconverter;
2
3 import android.os.AsyncTask;
4 import android.os.Bundle;
5 import android.support.v7.app.AppCompatActivity;
6 import android.view.View;
7 import android.widget.Button;
8 import android.widget.EditText;
9 import android.widget.TextView;
10 import android.widget.Toast;
11
12 import java.io.BufferedReader;
13 import java.io.IOException;
14 import java.io.InputStream;
15 import java.io.InputStreamReader;
16 import java.net.HttpURLConnection;
17 import java.net.MalformedURLException;
18 import java.net.URL;
19
20 public class MainActivity extends AppCompatActivity {
21     //Declare some variables
22     private EditText editText01;
23     private Button bnt01;
24     private TextView textView01;
25     private String usd;
26
27     //URL used to extract real-time currency exchange rates, using JSON, for the Petrol-AirCrafftCarrier-Federal-Reserve-Note
28     String json_url = "https://api.fixer.io/latest?base=USD";
29
30     //JSON Object extracted from website
31     String JSON_String;
32
33
34
35
36 @Override
37 protected void onCreate(Bundle savedInstanceState) {
38     super.onCreate(savedInstanceState);
39     setContentView(R.layout.activity_main);
40     //cast the variables to their ids
41     editText01 = findViewById(R.id.EditText01);
42     bnt01 = findViewById(R.id.bnt);
43     textView01 = findViewById(R.id.Yen);
44 }
```

3. Let us begin. Move the cursor into the top of the **MainActivity** Class and declare two new variables of the String Class, below the variable **usd** but above the **onCreate(...)** method.

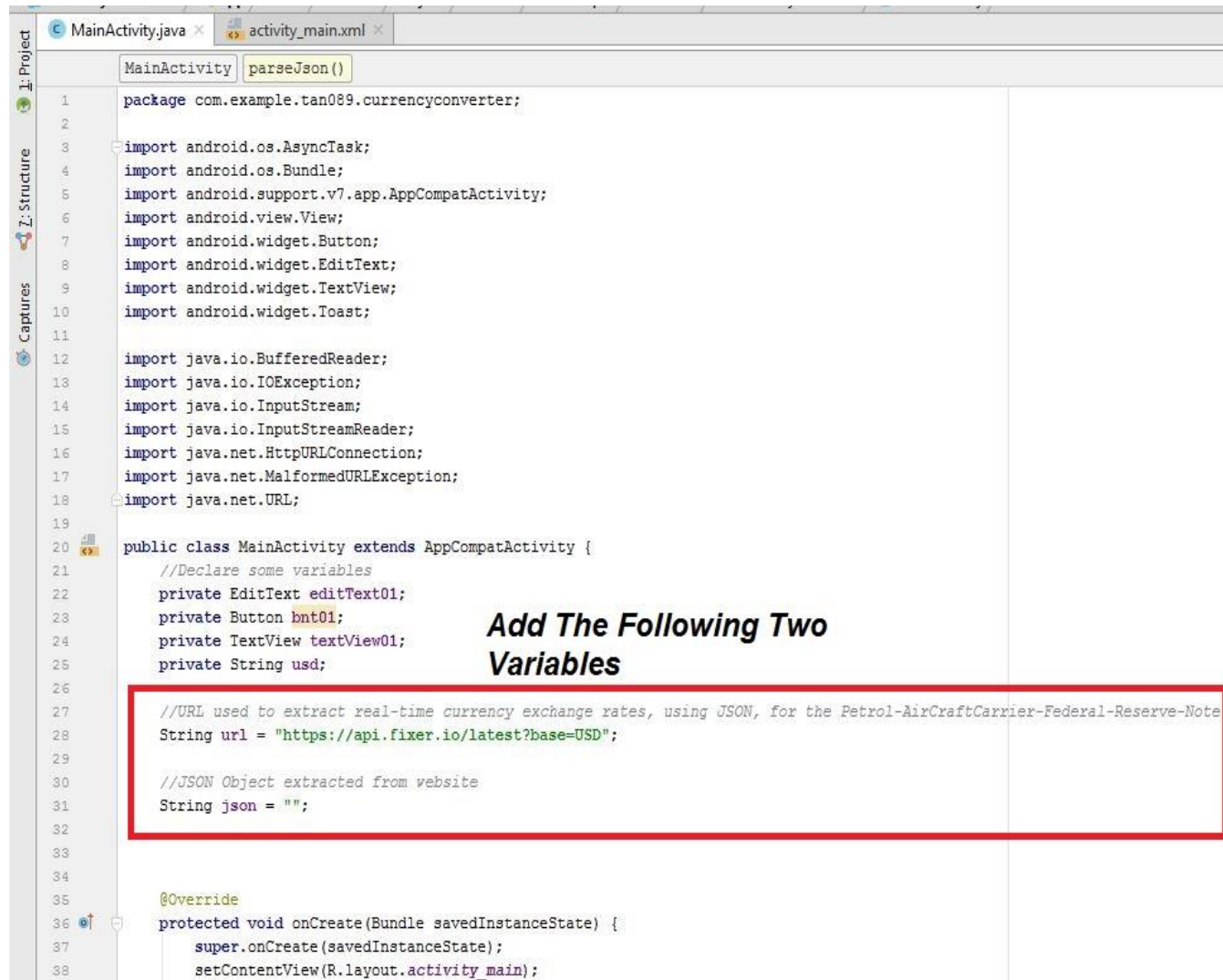
These two variables are **url** and **json**.

The **url** variable will store the URL for the website that we are using to extract a JSON object. (JSON is an acronym that stands for Java Script Object Notation)

For this tutorial, the URL we are using is <https://api.fixer.io/latest?base=USD>

The **url** variable will be initialized to this URL in the form of a String.

The second variable, **json**, will be initialized as an empty string. We will use the string **json** later within our code, in order to store the JSON object that will be extracted from the website. This extraction process will be done within the **Asynchronous Task**. (The JSON object extracted will be in the form of a string and it will be the most up to date currency conversion value for USD (U.S. Dollars)).



```
1 package com.example.tan089.currencyconverter;
2
3 import android.os.AsyncTask;
4 import android.os.Bundle;
5 import android.support.v7.app.AppCompatActivity;
6 import android.view.View;
7 import android.widget.Button;
8 import android.widget.EditText;
9 import android.widget.TextView;
10 import android.widget.Toast;
11
12 import java.io.BufferedReader;
13 import java.io.IOException;
14 import java.io.InputStream;
15 import java.io.InputStreamReader;
16 import java.net.HttpURLConnection;
17 import java.net.MalformedURLException;
18 import java.net.URL;
19
20 public class MainActivity extends AppCompatActivity {
21     //Declare some variables
22     private EditText editText01;
23     private Button bnt01;
24     private TextView textView01;
25     private String usd;
26
27     //URL used to extract real-time currency exchange rates, using JSON, for the Petrol-AirCraftCarrier-Federal-Reserve-Note
28     String url = "https://api.fixer.io/latest?base=USD";
29
30     //JSON Object extracted from website
31     String json = "";
32
33
34
35
36 @Override
37 protected void onCreate(Bundle savedInstanceState) {
38     super.onCreate(savedInstanceState);
39     setContentView(R.layout.activity_main);
40 }
```

Add The Following Two Variables

4. In the Click Event method, `bnt01.setOnClickListener(...`, let us prepare by *converting* the String value in the variable `json` to a type Double. We will do this by declaring and initializing a variable of type Double, called **convert**.

Convert the JSON string to type double and initialize the value of type double to the variable **convert**, all at once like so:

```
double convert = Double.parseDouble(json);
```

```
//Click event
bnt01.setOnClickListener(new View.OnClickListener() { Initialize variable,
    @Override "convert", here
    public void onClick(View convertToYen) {
        //obtain the latest currency-conversion value from URL, convert from String to Double
        double convert = Double.parseDouble(json);

        //convert user's input to string
        usd = editText01.getText().toString();
        //if-else statement to make sure user cannot leave the EditText blank
        if (usd.equals("")){
            textView01.setText("This field cannot be blank!");
        } else {
            //Convert string to double
            Double dInputs = Double.parseDouble(usd);

            //Convert function, using the value obtained from the URL, multiply it by the user input
            Double result = dInputs * convert;
            //Display the result
            textView01.setText("$" + usd + " = " + "¥"+String.format("%.2f", result));
            //clear the edittext after clicking
            editText01.setText("");
        }
    }
});
```

The variable, **convert**, replaces the value of type double in the initialization of the variable, **result**, from the tutorial in Week 1.

Let us recall that the variable `json` is an empty string as of now, but later, it will be assigned a value of type string from the website that will be accessed in the **Asynchronous Task**. This step is in preparation for when we obtain this JSON object.

5. Now, we will begin to write the class that will contain the **Asynchronous Task** used to extract the JSON object from the currency-conversion web Application Programming Interface, api.fixer.io

We will name the class ***BackgroundTask*** and write it below the Click Event method, **bnt01.setOnClickListener(...**

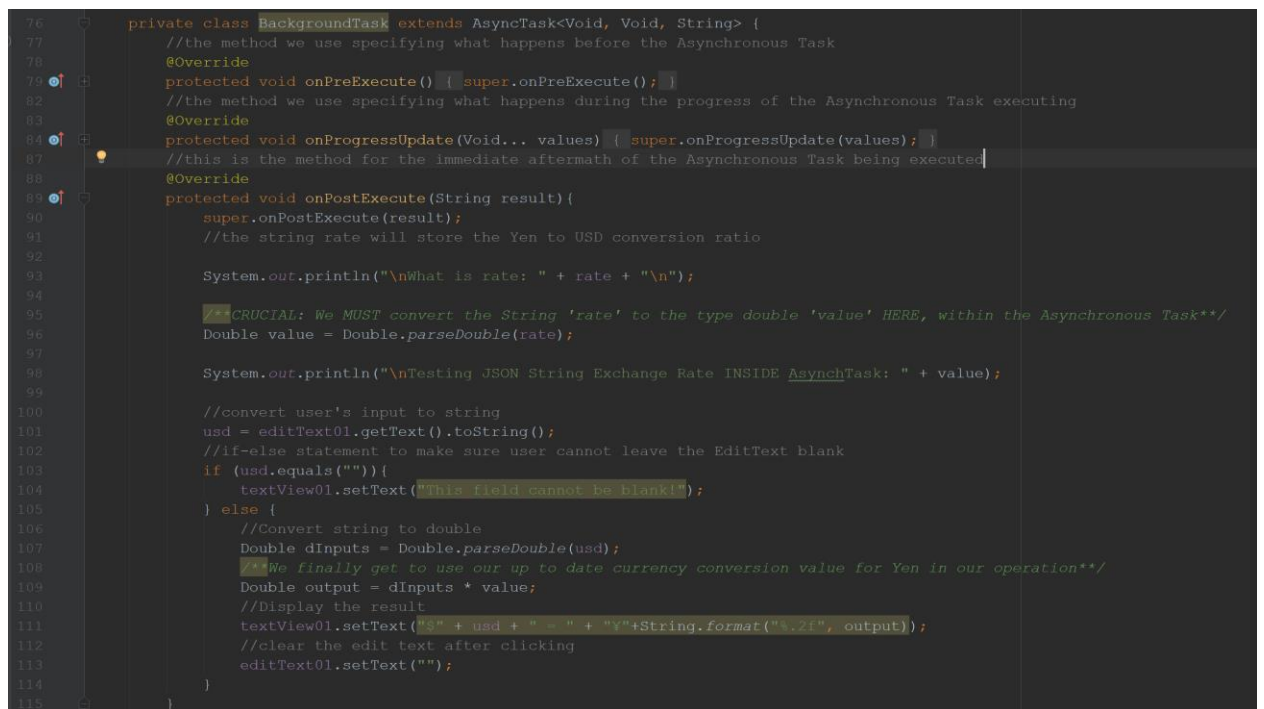
```
38 //Click event
39 bnt01.setOnClickListener(new View.OnClickListener() {
40     @Override
41     public void onClick(View convertToYen) {
42         //obtain the latest currency-conversion value from URL, convert the JSON from String to Double
43         double convert = Double.parseDouble(json);
44
45         //convert user's input to string
46         usd = editText01.getText().toString();
47         //if-else statement to make sure user cannot leave the EditText blank
48         if (usd.equals("")){
49             textView01.setText("This field cannot be blank!");
50         } else {
51             //Convert string to double
52             Double dInputs = Double.parseDouble(usd);
53
54             //Convert function, using the value obtained from the URL, multiply it by the user input
55             Double result = dInputs * convert;
56             //Display the result
57             textView01.setText("$" + usd + " = " + "¥" + String.format("%.2f", result));
58             //clear the editText after clicking
59             editText01.setText("");
60         }
61     }
62 }
63 }
```

Begin to write the BackgroundTask Class Below. The nested method, doInBackground(String... params) will be defined more later

```
65 private class BackgroundTask extends AsyncTask<String, Void, String> {
66     @Override
67     protected String doInBackground(String... params) {
68
69         //Stub: More code will define this nested method in the BackgroundTask class later
70         //The code written in this method will be the primary instructions carried out within the Asynchronous Task
71
72         return null;
73     }
74 }
75 }
```

The class, BackgroundTask, will contain most of the instructions for the app's Asynchronous Task

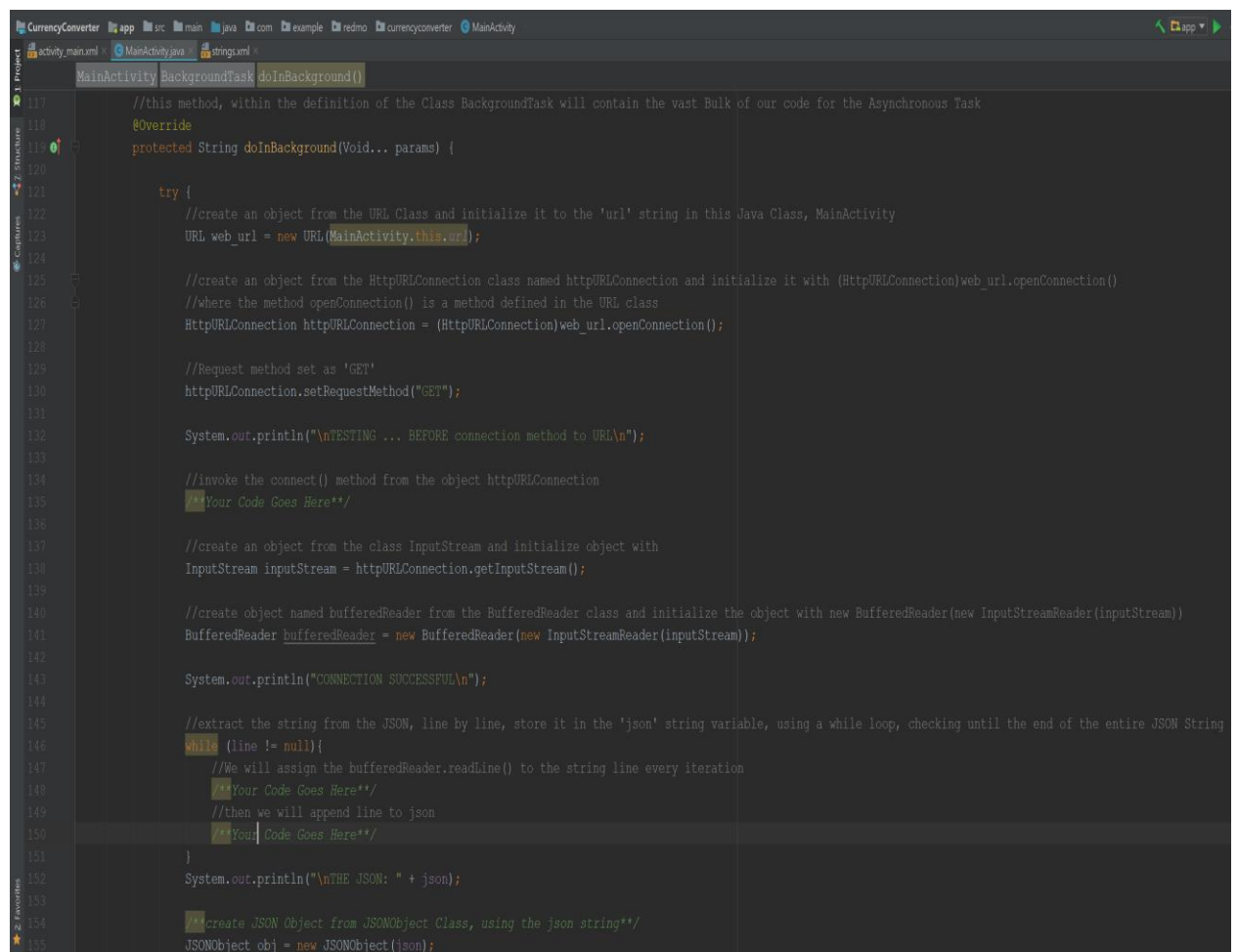
- Take the logical code for currency conversion from Tan's code, then cut and paste it into the `onPostExecute(String result)` method



Change the name 'result' of the variable that is of type string to the name 'output'

- The following screen shot shows the majority of code you will need for your BackgroundTask class, that extends AsyncTask. You will need to create objects from the following classes: URL, HttpURLConnection, InputStream, BufferedReader, and JSONObject.

You will implement over several methods from the objects of these classes in order to retrieve the JSON, up-to-date currency conversion rate. Again, you are more than welcome to use the Volley library, supported by Android Studio, in place of using the verbose classes for the Asynchronous Task.



```
117 //this method, within the definition of the Class BackgroundTask will contain the vast Bulk of our code for the Asynchronous Task
118 @Override
119 protected String doInBackground(Void... params) {
120
121     try {
122         //create an object from the URL Class and initialize it to the 'url' string in this Java Class, MainActivity
123         URL web_url = new URL(MainActivity.this.url);
124
125         //create an object from the HttpURLConnection class named httpURLConnection and initialize it with (HttpURLConnection)web_url.openConnection()
126         //where the method openConnection() is a method defined in the URL class
127         HttpURLConnection httpURLConnection = (HttpURLConnection)web_url.openConnection();
128
129         //Request method set as 'GET'
130         httpURLConnection.setRequestMethod("GET");
131
132         System.out.println("\nTESTING ... BEFORE connection method to URL\n");
133
134         //invokes the connect() method from the object httpURLConnection
135         /**Your Code Goes Here**/
136
137         //create an object from the class InputStream and initialize object with
138         InputStream inputStream = httpURLConnection.getInputStream();
139
140         //create object named bufferedReader from the BufferedReader class and initialize the object with new BufferedReader(new InputStreamReader(inputStream))
141         BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
142
143         System.out.println("CONNECTION SUCCESSFUL\n");
144
145         //extract the string from the JSON, line by line, store it in the 'json' string variable, using a while loop, checking until the end of the entire JSON String
146         while (line != null){
147             //We will assign the bufferedReader.readLine() to the string line every iteration
148             /**Your Code Goes Here**/
149             //then we will append line to json
150             /**Your Code Goes Here**/
151         }
152         System.out.println("\nTHE JSON: " + json);
153
154         //create JSON Object from JSONObject Class, using the json string**/
155         JSONObject obj = new JSONObject(json);
```

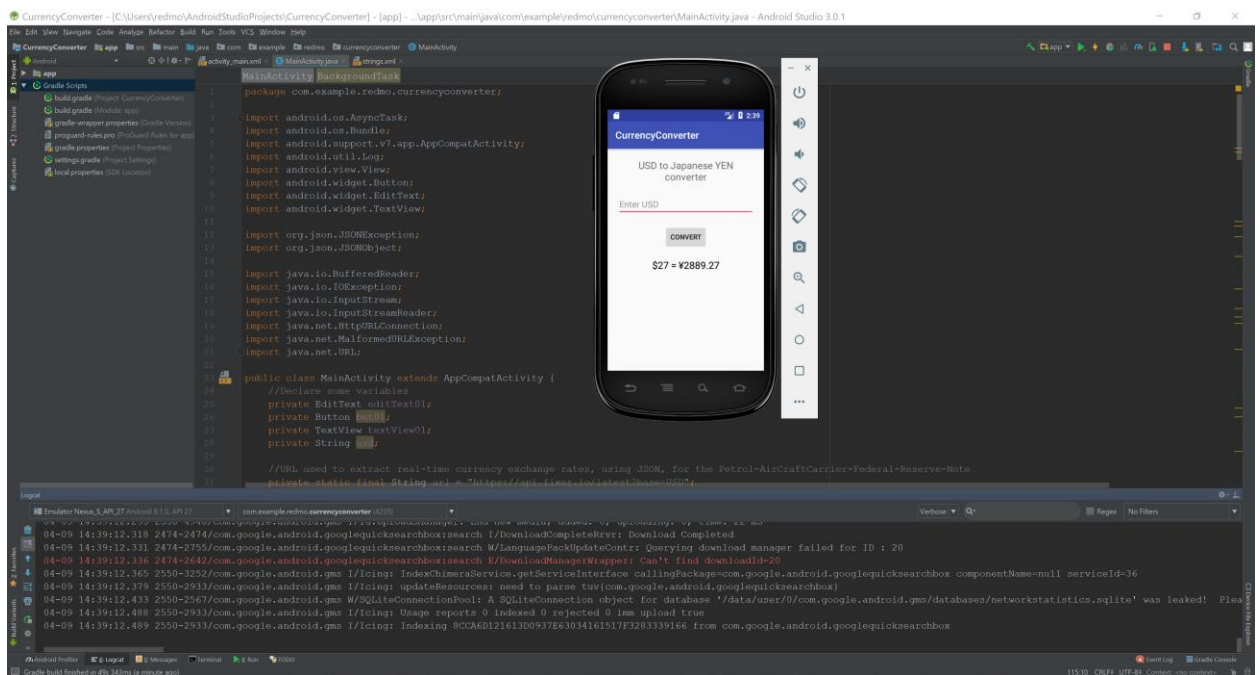
I leave merely 3 lines blank, for you to fill in your code. Each are very basic instantiations of methods called from objects and the basic use of the assignment operator. (I will go over these 3 lines in the Laboratory Tutorial);

8. The following screen shot shows what should be inside the end of the try, catch, and throw block of the BackgroundTask Class. Also, it shows how to use the JSONObject class to instantiate an object to acquire the web-page label data “rates”.

In addition, it is necessary to convert this data to type String first, before using the data for any more tasks.

```
157 //create second JSON Object that will contain a nested JSON Object within the FIRST JSON Object created
158 JSONObject objRate = obj.getJSONObject("rates");
159
160 //use the second JSON Object created and use the get(String ...) method
161 //We will put "JPY" as the argument in the parameter of the get(String ...) method in order to get the exchange rate for Yen
162 //The logic code for currency conversion MUST GO INTO the onPostExecute() Method within AsyncTask
163
164 rate = objRate.get("JPY").toString();
165
166
167
168 } catch (MalformedURLException e) {
169     e.printStackTrace();
170 } catch (IOException e) {
171     e.printStackTrace();
172 } catch (JSONException e) {
173     Log.e(tag, "MYAPP", msg, "unexpected JSON exception", e);
174     System.exit(status:1);
175 }
176 return null;
177 }
178 }
179 }
180 }
```

9. Finally, this snapshot shows what your emulator should look like after you have successfully implemented Asynchronous task.



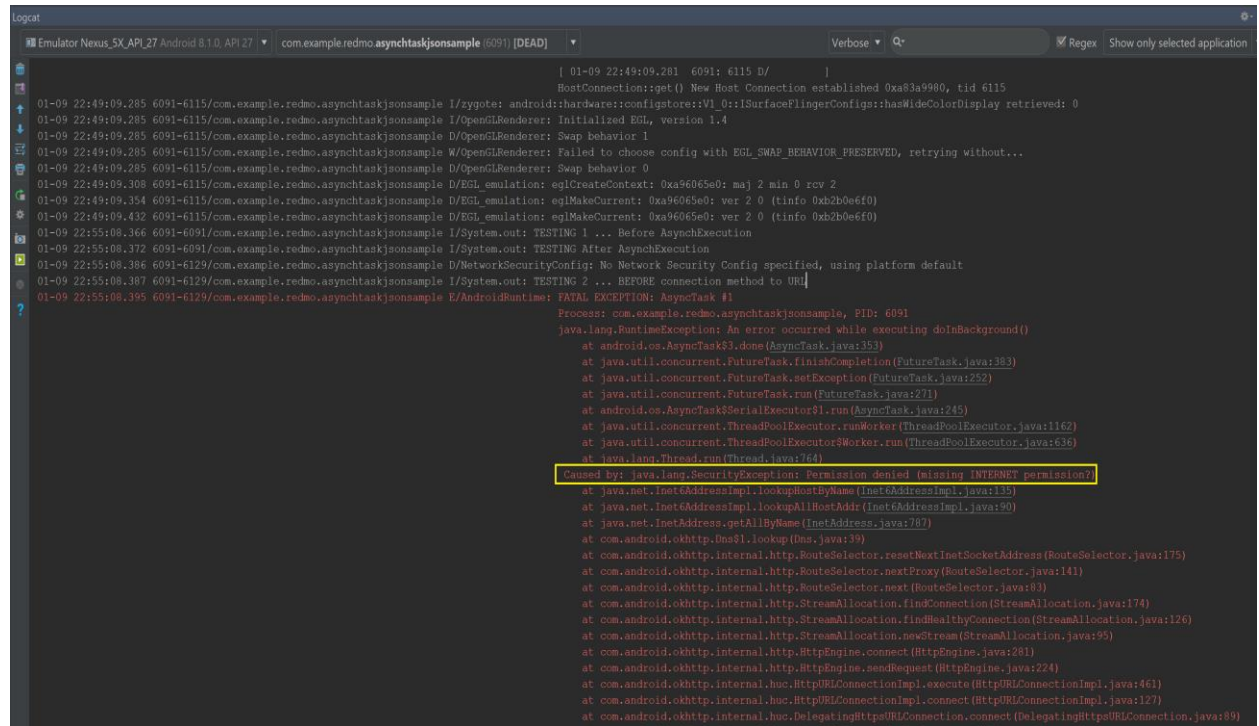
10. The following is URL JSON data from the website: api.fixer.io

base: "USD"
date: "2018-01-08"
rates:

AUD:	1.277
BGN:	1.6335
BRL:	3.2427
CAD:	1.2428
CHF:	0.97795
CNY:	6.4993
CZK:	21.318
DKK:	6.2196
GBP:	0.73844
HKD:	7.8223
HRK:	6.2157
HUF:	257.95
IDR:	13434
ILS:	3.4417
INR:	63.508
JPY:	113.04
KRW:	1067.3
MXN:	19.278
MYR:	4.0031
NOK:	8.0834
NZD:	1.395
PHP:	50.192
PLN:	3.4797
RON:	3.8592
RUB:	57.109
SEK:	8.1989
SGD:	1.3327
THB:	32.21
TRY:	3.749
ZAR:	12.438
EUR:	0.83521

11. The Missing Internet Permission

Depending on the configuration of your Manifest.xml file in Android Studio, you will likely not have the proper Internet Permission configured. Do not be surprised if you get this error the first time you build and run the app with Asynchronous Task:



```
Logcat
Emulator Nexus_5X_API_27 Android 8.1.0, API 27 | com.example.redmo.asynctaskjsonsample [DEAD] | Verbose | Q | Regex | Show only selected application

01-09 22:49:09.285 6091-6115/com.example.redmo.asynctaskjsonsample I/zygote: android.hardware.configstore:V1_0:ISurfaceFlingerConfigs:hasWideColorDisplay retrieved: 0
01-09 22:49:09.285 6091-6115/com.example.redmo.asynctaskjsonsample I/OpenGLRenderer: Initialized EGL, version 1.4
01-09 22:49:09.285 6091-6115/com.example.redmo.asynctaskjsonsample D/OpenGLRenderer: Swap behavior 1
01-09 22:49:09.285 6091-6115/com.example.redmo.asynctaskjsonsample W/OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...
01-09 22:49:09.308 6091-6115/com.example.redmo.asynctaskjsonsample D/OpenGLRenderer: Swap behavior 0
01-09 22:49:09.354 6091-6115/com.example.redmo.asynctaskjsonsample D/EGL_emulation: eglCreateContext: 0xa96065e0; maj 2 min 0 rev 2
01-09 22:49:09.432 6091-6115/com.example.redmo.asynctaskjsonsample D/EGL_emulation: eglMakeCurrent: 0xa96065e0; ver 2 0 (info 0xb2b0e6f0)
01-09 22:49:09.432 6091-6115/com.example.redmo.asynctaskjsonsample D/EGL_emulation: eglMakeCurrent: 0xa96065e0; ver 2 0 (info 0xb2b0e6f0)
01-09 22:55:08.366 6091-6091/com.example.redmo.asynctaskjsonsample I/System.out: TESTING 1 ... Before AsyncExecution
01-09 22:55:08.372 6091-6091/com.example.redmo.asynctaskjsonsample I/System.out: TESTING After AsyncExecution
01-09 22:55:08.386 6091-6129/com.example.redmo.asynctaskjsonsample D/NetworkSecurityConfig: No Network Security Config specified, using platform default
01-09 22:55:08.387 6091-6129/com.example.redmo.asynctaskjsonsample I/System.out: TESTING 2 ... BEFORE connection method to URL
01-09 22:55:08.395 6091-6129/com.example.redmo.asynctaskjsonsample E/AndroidRuntime: FATAL EXCEPTION: AsyncTask #1
    Process: com.example.redmo.asynctaskjsonsample, PID: 6091
    java.lang.RuntimeException: An error occurred while executing doInBackground()
        at android.os.AsyncTask$3.done(AsyncTask.java:353)
        at java.util.concurrent.FutureTask.finishCompletion(FutureTask.java:383)
        at java.util.concurrent.FutureTask.setException(FutureTask.java:252)
        at java.util.concurrent.FutureTask.run(FutureTask.java:271)
        at android.os.AsyncTask$SerialExecutor$1.run(AsyncTask.java:245)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1162)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:636)
        at java.lang.Thread.run(Thread.java:764)
    Caused by: java.lang.SecurityException: Permission Denied (missing INTERNET permission?)
        at java.net.InetAddressImpl.lookupHostName(InetAddressImpl.java:135)
        at java.net.InetAddressImpl.lookupAllHostAddr(InetAddressImpl.java:90)
        at java.net.InetAddress.getAllByNames(InetAddress.java:787)
        at com.android.okhttp.Dns$1.lookup(Dns.java:39)
        at com.android.okhttp.internal.http.RouteSelector.resetNextInetSocketAddress(RouteSelector.java:175)
        at com.android.okhttp.internal.http.RouteSelector.nextProxy(RouteSelector.java:141)
        at com.android.okhttp.internal.http.RouteSelector.next(RouteSelector.java:83)
        at com.android.okhttp.internal.http.StreamAllocation.findConnection(StreamAllocation.java:174)
        at com.android.okhttp.internal.http.StreamAllocation.findHealthyConnection(StreamAllocation.java:126)
        at com.android.okhttp.internal.http.StreamAllocation.newStream(StreamAllocation.java:95)
        at com.android.okhttp.internal.http.HttpEngine.connect(HttpEngine.java:281)
        at com.android.okhttp.internal.http.HttpEngine.sendRequest(HttpEngine.java:224)
        at com.android.okhttp.internal.huc.HttpURLConnectionImpl.execute(HttpURLConnectionImpl.java:461)
        at com.android.okhttp.internal.huc.HttpURLConnectionImpl.connect(HttpURLConnectionImpl.java:127)
        at com.android.okhttp.internal.huc.DelegatingHttpsURLConnection.connect(DelegatingHttpsURLConnection.java:89)
```

To fix this error, you will need to go into you Manifest.xml file in Android Studio, and then copy-paste the proper permission code for your app to access the Internet and obtain the JSON data for the currency conversion.

You can get this .xml line for your Manifest file by simply doing a Google search with terms such as “Missing Internet Permission Android” and the answer will come right up. This is a very simple, common issue when doing beginner Android Studio programming, so it is very easy to find, thankfully.